GASFacCop Documentation

May 19, 2023

Contents

1 Authorship

The GASFacCop library is an adaptation of the code provided by Dong Hwan Oh and Andrew Patton in their paper on factor copulas with GAS dynamics. The original code was modified and extended to improve usability and add new features. The `GASFacCop.m` script, `generate_copula_dynamic.m`, `generate_copula_static.m`, `get_bounds.m`, the graphical interface, and the documentation were created by Victor Cozer, Emilio Ferrante e Francesco Girardi. Some other functions were adapted from different authors, for which we provide due credit in the respective function description.

2 Library Description

The GASFacCop library is a MATLAB-based tool that allows users to estimate a factor copula with GAS (Generalized Autoregressive Score) dynamics. This documentation provides an overview of the GASFacCop library, including its usage, functionalities, and the underlying code structure.

The GASFacCop library provides a set of functions and scripts for estimating a factor copula with GAS dynamics. The main purpose of the library is to fit a factor copula model to multivariate data and generate simulations from the fitted copula. The library supports various options for selecting distributions for the latent variables and the factors, as well as different dependence structures.

The GASFacCop library is an adaptation of the code provided by Oh and Patton in their paper on factor copulas with GAS dynamics. The original code has been modified and extended to enhance its usability and provide a user-friendly interface. In addition, new functions and scripts have been added to support dynamic and static factor copula models, simulation capabilities, and graphical interfaces.

3 Installation

To use the GASFacCop library, follow these steps:

1. Download the GASFacCop package
2. Unzip the folder containing the package
3. Open MATLAB on your computer
4. Open the unzipped folder in you Matlab enviroment
5. Use the graphical interface in the file "`GASFacCop.m`"
6. You may be asked to install the required dependencies to run the package.

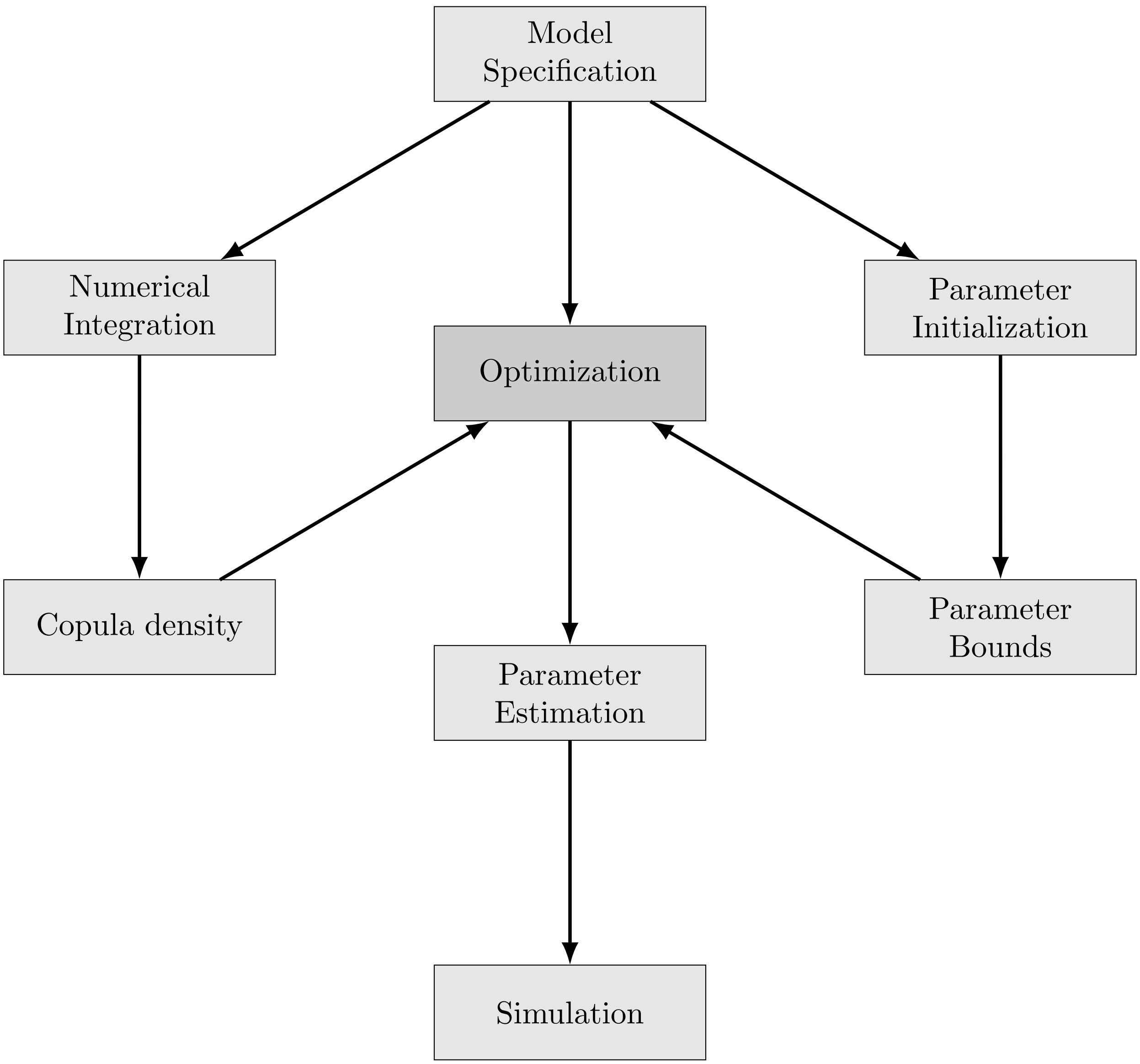

Figure 1: Estimation and Simulation Procedure

4 Usage

To fit a factor copula model using the GASFacCop library, follow these steps:

4.1 Input Data

1. Prepare your dataset in MATLAB, ensuring that it is in the required format. The dataset should be a matrix where each column represents a time series of a variable. The marginal distributions of the variables should be uniformly distributed, i.e., all elements should be in the range $[0, 1]$.
2. Assign the relevant columns of the dataset to the variable `copula_data` in the MATLAB workspace.
3. Define the grouping of variables (if applicable) using the `group_code` variable. This is required for block dependence modeling. If not using block dependence, this is step is not required.

4.2 Running the GASFacCop Script

1. Run the GASFacCop script in MATLAB.
2. A graphical interface will appear, allowing you to choose various specifications for the factor copula model.
3. Select the desired options from the dropdown menus and input fields in the graphical interface.
4. Click the "Submit" button to start the fitting process.
5. Depending on the chosen options and the size of the dataset, the estimation process may take some time to complete. Monitor the progress in the MATLAB console.

4.3 Simulation (Optional)

The GASFacCop library provides the option to simulate the fitted copula after the estimation process. Once the estimation is complete, the user is prompted with a dialog asking whether they want to simulate the copula. If the user chooses to simulate, they can input the number of days and the number of simulations for the desired copula simulations.

The simulation process differs based on the selected model type:

- **Static Model**: For the static model, the function `generate_copula_static` is used to generate copula simulations. It takes as input the number of simulations, the number of days, the model parameters, the initial lambda values, and the group assignments. The function returns the simulated copula data.

- **Dynamic Model**: For the dynamic model, the function `generate_copula_dynamic` is used to generate copula simulations. It takes as input the number of simulations, the number of days, the model parameters, the omega values, the initial lambda values, and the group assignments. The function returns the simulated copula data.

Once the simulation process is complete, the simulated copula data can be saved to a file for further analysis or visualization.

4.4 Graphical Interface

The GASFacCop.m script includes a graphical user interface (GUI) that allows users to select the options for the factor copula model. It consists of dropdown menus and input fields where users can choose the factor distribution, innovation distribution, model type, dependence structure, and the number of iterations. The GUI provides a user-friendly way to interact with the library, initiate the fitting process and simulate according to the fitted model.

5 Code Structure

The GASFacCop library consists of several MATLAB functions and scripts that work together to estimate the factor copula model with GAS dynamics. The main components of the library are as follows:

5.1 GASFacCop.m

The GASFacCop.m script is the main interface that users interact with. It provides a graphical user interface (GUI) where users can select various options for the factor copula model, such as the factor distribution, innovation distribution, model type (static or dynamic), and dependence structure. It also handles the input data, calls the appropriate estimation functions, and initiates the simulation process if requested.

5.2 get_bounds.m

The `get_bounds.m` function calculates the parameter bounds for the estimation process. It takes as input the number of factors and returns the lower and upper bounds for the factor loadings and the shape parameters. Throughout all the estimation and simulation procedure, we use a Skew Student's t pdf as a building block. For example, in order to use a Gaussian distribution, it sufficient to use the Skew Student's t with infinite degrees of freedom and zero asymmetry parameter.

5.3 Function: GASFacCop_G

Description: This function is used to estimate a factor copula with GAS dynamics when the total number of groups G is strictly less than the number of variables N.

Usage:

```
[theta_NMLE, lam] = GASFacCop_G(copula_data, factor_dist,
    innov_dist, model, max_iter, group_code)
```

Input:

- `copula_data`: A matrix of size (T x N) representing the copula data.
- `factor_dist`: The distribution of the factors.
- `innov_dist`: The distribution of the innovations.
- `model`: The type of model.
- `max_iter`: The maximum number of iterations for optimization.
- `group_code`: A vector of size (N x 1) representing the group codes.

Output:

- `theta_NMLE`: The estimated parameters.
- `lam`: The estimated factor loadings.

Details: The function performs the following steps:

- Initializes the number of groups `N_group` based on the maximum value of `group_code`.
- Sets up the nodes and weights for Gauss-Legendre quadrature, which are used for computing the copula density.
- Estimates the initial factor loadings `loading_ini` based on the correlations of the first 65 observations.
- Sets up the optimization options for the estimation process.
- Defines the upper and lower bounds for the optimization based on the `factor_dist`, `innov_dist`, and `model`.
- Initiates the estimation of the factor copula with GAS recursion using the `fminsearchbnd` function.
- If needed, the function can output the estimated parameters `theta_NMLE`, the objective function value `fobj`, the exit flag `exitflag`, and the optimization output `output1`.

- It can also calculate additional output measures such as the objective function value `obj`, log-likelihood `LL`, lambda `lam`, log lambda `log_lam`, and the score `s`.
- Returns the estimated parameters, lambda values, and estimation time.

The `GASFacCop_G` function relies on the optimization routine `fminsearchbnd` and the log-likelihood calculation function `LLSum_GASFacCop_G` to estimate the parameters and perform the likelihood maximization.

5.4 Function: GASFacCop_N

Description: This function is used to estimate a factor copula with GAS dynamics when the total number of groups G is equal to the number of variables N.

Usage:

```
[theta_NMLE, lam] = GASFacCop_N(copula_data, factor_dist,
    innov_dist, model, max_iter)
```

Input:

- `copula_data`: A matrix of size (T x N) representing the copula data.
- `factor_dist`: The distribution of the factors.
- `innov_dist`: The distribution of the innovations.
- `model`: The type of model.
- `max_iter`: The maximum number of iterations for optimization.

Output:

- `theta_NMLE`: The estimated parameters.
- `lam`: The estimated factor loadings.

Details: The function `GASFacCop_N` performs the following steps:

- Initializes the nodes and weights for Gauss-Legendre quadrature, which are used for computing the copula density.
- Estimates the initial values for the lambda parameters `lam_ini` based on the Spearman correlations of the first 65 observations and the full dataset.
- Defines the bounds for the parameter optimization based on the factor_dist, innov_dist, and `model`.
- Sets up the optimization options for the estimation process.
- Initiates the estimation of the factor copula with GAS recursion using the `fminsearchbnd` function.

- Outputs the estimated parameters theta_NMLE and the lambda values am.
- Calculates the estimation time.

The GASFacCop_N function relies on the optimization routine fminsearchbnd and the log-likelihood calculation function LLSum_GASFacCop_N to estimate the parameters and perform the likelihood maximization.

5.5 Function: LLSum_GASFacCop_G

Description: This function computes the sum of (negative) log likelihoods of factor copula with GAS recursion.

Usage:

```
[out, LL, lam, log_lam, s ] = LLSum_GASFacCop_G(theta,
    data_u, GLweight, group_code, lam_ini)
```

Input:

- theta: A vector of parameters.
- data_u: A matrix of size (T x N) representing the data to be modeled with the copula.
- GLweight: A matrix of nodes and weights for Gauss-Legendre quadrature.
- group_code: A vector of size (N x 1) representing the group codes.
- lam_ini: A vector of size (Ngroup x 1) representing the initial factor loadings at t=1.

Output:

- out: A scalar representing the sum of (negative) log-likelihoods.
- LL: A vector of size (T x 1) representing the log-likelihoods at each time t.
- lam: A matrix of size (T x N) representing the time-varying factor loadings.
- log_lam: A matrix of size (T x N) representing the log of factor loadings.
- s: A matrix of size (T x N) representing the scores.

Details:

The function LLSum_GASFacCop_G calculates the log-likelihood function value, lambda values, log lambda values, and the score for the factor copula with GAS dynamics when the total number of groups Ngroup is less than the number of variables N. It performs the following steps:

- Extracts the parameters from the input theta vector.

- Initializes the log-likelihood LL and the score s to zero.
- Loops over each group i from 1 to Ngroup.
  - Filters the data for the current group i based on the group code group_code.
  - Calculates the factor loadings lam_t for the current group using the previous loadings loading_ini and the parameters theta.
  - Filters the innovations for the current group.
  - Calculates the conditional covariance matrix Sigma_t for the current group.
  - Calculates the log-likelihood contribution ll_t for the current group.
  - Updates the overall log-likelihood LL by adding ll_t.
  - Updates the overall score s by adding the derivative of ll_t with respect to theta.
- Calculates the lambda values lam and the log lambda values log_lam based on the filtered data.

The LLSum_GASFacCop_G function relies on the functions factor_copula_GAS, Q_GAS, eta_GAS, GASpdf, and X_pdf_eval to calculate the log-likelihood contributions, the conditional covariance matrix, and the density evaluations.

5.6 Function: LLSum_GASFacCop_N

Description: This function computes the sum of (negative) log likelihoods of the factor copula (skew t - t) with GAS recursion.

Usage:

```
[out, LL, lam, log_lam, s ] = LLSum_GASFacCop_N(theta,
    data_u, GLweight, lam_bar, lam_ini)
```

Input:

- theta: A vector of parameters [alpha, beta, nuinv_z, nuinv_eps, psi_z].
- data_u: A matrix of size (T x N) representing the data to be modeled with the copula.
- GLweight: A matrix of nodes and weights for Gauss-Legendre quadrature.
- lam_bar: A vector of size (N x 1) representing the average factor loadings given from the separate stage.
- lam_ini: A vector of size (N x 1) representing the factor loadings at t=1.

Output:

- `out`: A scalar representing the sum of (negative) log-likelihoods of the factor copula evaluated at each time t.
- `LL`: A vector of size (T x 1) representing the log-likelihoods of the factor copula evaluated at each time t.
- `lam`: A matrix of size (T x N) representing the time-varying factor loadings.
- `log_lam`: A matrix of size (T x N) representing the log of factor loadings.
- `s`: A matrix of size (T x N) representing the scores.

Details: The `LLSum_GASFacCop_N` function computes the log-likelihood function value, lambda values, log lambda values, and the score for the factor copula with GAS recursion when the total number of groups `Ngroup` is equal to the number of variables `N`. It performs the following steps:

- Extracts the parameters from the input `theta` vector.
- Initializes the log-likelihood `LL` and the score `s` to zero.
- Loops over each time period `tt` from 1 to `TT`.
  - Checks if `tt` is not equal to 1.
    * Updates the log lambda values `log_lam` and the lambda values `lam` based on the previous values and the parameters `omega`, `alpha`, `beta`.
  - Sets lower and upper bounds for the lambda values `lam` to ensure they are within the desired range.
  - Calls the `LL_GASFacCop_G` function to calculate the log-likelihood contribution `L_temp` and the score vector `N_derivative`.
  - Updates the overall log-likelihood `LL` by adding `L_temp`.
  - Updates the overall score `s` by multiplying `N_derivative` with `lam`.
- Calculates the negative sum of log-likelihoods `out` by summing the negative values of `LL`.

5.7 Function: LL_GASFacCop_G

Description: This function computes the log likelihood of factor copula evaluated at time t. It also generates the numerical derivatives of log copula density with respect to each group's factor loading at time t.

Usage:

```
[LL_t, N_derivative] = LL_GASFacCop_G(theta_t, u_t, GLweight
    , Gcdf, Gpdf, x_grid, lam_grid, group_code, epsi)
```

Input:

- `theta_t`: A vector of parameters at time t.
- `u_t`: A vector of size (N x 1) representing the data at time t.
- `GLweight`: A matrix of nodes and weights for Gauss-Legendre quadrature.
- `Gcdf`: A matrix of size (Num_x_grid x Num_lam_grid) representing the marginal cdfs of the skew t-t factor model.
- `Gpdf`: A matrix of size (Num_x_grid x Num_lam_grid) representing the marginal pdfs of the skew t-t factor model.
- `x_grid`: A vector representing the x grid for Gcdf and Gpdf.
- `lam_grid`: A vector representing the factor loading grid for Gcdf and Gpdf.
- `group_code`: A vector of size (N x 1) representing the group codes.
- `epsi`: A scalar representing the step size for numerical derivative calculation.

Output:

- `LL_t`: A scalar representing the log-likelihood at time t.
- `N_derivative`: A vector of size (N_group x 1) representing the numerical derivatives of log copula density.

Details:

The function `LL_GASFacCop_G` calculates the log-likelihood of the factor copula evaluated at time `t` and generates the numerical derivatives of the log copula density with respect to each group's factor loading at time `t`. It performs the following steps:

- Checks the dimensions of the input variables and ensures consistency.
- Initializes empty arrays for storing intermediate results.
- Loops over each group `i` from 1 to the maximum group code `max(group_code)`.
  - Filters the data for the current group `i` based on the group code `group_code`.
  - Extracts the group-specific parameters from the input `theta_t` vector.
  - Calculates the inverse of the copula function `Ginv_u_t` and `Ginv_u_t_ateps` for the current group.
  - Calculates the denominator terms `deno_t` and `deno_t_ateps` for the current group.
- Calculates the numerator term `numer_t` using the joint copula density function.

- Computes the log-likelihood value LL_t by taking the logarithm of the first element of numer_t and subtracting the sum of the logarithms of deno_t.
- If requested, calculates the numerical derivative of the log copula density with respect to each group's factor loading by looping over each group i.
  - Modifies the denominator term deno_temp by replacing the elements corresponding to the current group with the values from deno_t_ateps.
  - Calculates the log-likelihood value LL_eps for the modified denominator term.
  - Computes the numerical derivative N_derivative(i) by taking the difference between LL_eps and LL_t divided by the step size epsi.
- Returns the log-likelihood value LL_t and the numerical derivatives N_derivative.

The LL_GASFacCop_G function relies on the functions X_inverse_cdf and X_pdf_eval to compute the inverse of the copula function and evaluate the copula density, respectively.

5.8 Function: generate_copula_dynamic1

Description: This function is used to generate data from a dynamic block-dependent factor copula.

Usage:

```
[U, lambda_mat] = generate_copula_dynamic1(T, dim, theta,
    omega, lambda_init, group_assign)
```

Input:

- T: The number of days.
- dim: The number of stocks.
- theta: A vector of parameters.
- omega: A vector of omega values.
- lambda_init: The initial lambda values.
- group_assign: The division into groups.

Output:

- U: A matrix representing the generated data.
- lambda_mat: A matrix representing the estimated factor loadings.

Output:

- U: A matrix of size representing the generated data.
- lambda_mat: A matrix representing the estimated factor loadings.

5.9 Function: GLquad

Description:

This function performs Gauss-Legendre quadrature on a univariate function over the interval $[a, b]$. The function `f_name(x)` must take a vector of values `x` at which it is to be evaluated.

Usage:

```
out1 = GLquad(f_name, a, b, n, varargin)
```

Input:

- `f_name`: A string representing the function name.
- `a`: The lower bound of the interval.
- `b`: The upper bound of the interval.
- `n`: The number of nodes to use in the quadrature.
- `varargin`: Additional arguments to pass to the function.

Output:

- `out1`: The estimated integral of the function `f_name` over the interval $[a, b]$.

5.10 Function: GLNodeWt

Description: This function calculates the nodes and weights for Gauss-Legendre quadrature of arbitrary order.

Usage:

```
[x,w] = GLNodeWt(n)
```

Input:

- `n`: The order of the quadrature rule.

Output:

- `x`: A vector of nodes.
- `w`: A vector of weights.

5.11 Function: X_cdf

Description: This function calculates the marginal cumulative distribution function (CDF) of X_i associated with a skew $t - t$ factor model.

Usage:

```
out1 = X_cdf(x, theta, GLweight)
```

Input:

- x: The value at which the CDF is evaluated.
- theta: A vector of parameters $[\lambda, \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$.
- GLweight: A matrix of nodes and weights for Gauss-Legendre quadrature.

Output:

- out1: The value of the marginal CDF at x.

5.12 Function: X_cdf_helper

Description: This function is a helper function used for the calculation of the integral in the marginal cumulative distribution function (CDF) of X_i.

Usage:

```
out1 = X_cdf_helper(u, x, theta)
```

Input:

- u: A $K \times 1$ vector of values of u (used by the numerical integration function).
- x: A scalar, the value of x at which the integral is evaluated.
- theta: A vector of parameters $[\lambda, \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$.

Output:

- out1: A $K \times 1$ vector, the value of the argument of the integral at each value of u.

5.13 Function: X_inverse_cdf

Description: This function is used to calculate the inverse cumulative distribution function (CDF) of x.

Usage:

```
x = X_inverse_cdf(q, theta, Gcdf, x_grid, lam_grid)
```

Input:

- q: A vector in $(0, 1)$, the values at which the inverse CDF is evaluated.
- theta: A vector of parameters $[factorloadings; \nu_z^{-1}; \nu_\epsilon^{-1}; \psi_z]$.
- Gcdf: A matrix of size [Num_x_grid by Num_lam_grid], the marginal CDFs of the skew t-t factor model at x and factor loading (λ).
- x_grid: A vector of x values at which Gcdf is evaluated.
- lam_grid: A vector of factor loading values at which Gcdf is evaluated.

Output:

- x: A vector, the values of the inverse CDF at q.

5.14 Function: X_joint_pdf

Description: This function is used to calculate the joint density of $[X_1, \ldots, X_N]$ associated with a factor model. It evaluates the joint densities at ($factor\ loading+$ $[0, \ldots, \epsilon_i, \ldots, 0], \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z$).

Usage:

```
out1 = X_joint_pdf(x, theta, GLweight, group_code, epsi)
```

Input:

- `x`: A matrix of size $[N \times 2]$ representing the values of x at which the joint density is evaluated.
  - 1st column: $G^{-1}(u)$ evaluated at $\theta = [factor\ loading,\ \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$
  - 2nd column: $G^{-1}(u)$ evaluated at $\theta = [(factor\ loading+[0, \ldots, \epsilon_i, \ldots, 0]), \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$
- `theta`: A vector of parameters $[factor\ loading,\ \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$.
- `GLweight`: A matrix of nodes and weights for Gauss-Legendre quadrature.
- `group_code`: An $N \times 1$ vector of group codes into which each firm is classified.
- `epsi`: A scalar, the step size for numerical derivative for score calculation.

Output:

- `out1`: An $[(1 + \#\ of\ group) \times 1]$ vector.
  - 1st element: The value of the joint PDF of skew t-t evaluated at $\theta = [factor\ loading,\ \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$.
  - $(i+1)$th elements: The value of the joint PDF of skew t-t evaluated at $\theta = [(factor\ loading + [0, \ldots, \epsilon_i, \ldots, 0]), \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$.

5.15 Function: X_marginals

Description: This function is used to calculate the marginal cumulative distribution function (CDF) and probability density function (PDF) of X_i evaluated at x and factor loading (lam) in a factor model.

It utilizes the functions `X_cdf` and `X_pdf` to calculate the CDF and PDF of X_i at specific values of `x_grid` and `lam_grid`.

Usage:

```
[Gcdf, Gpdf] = X_marginals(theta, GLweight, x_grid, lam_grid
    )
```

Input:

- `theta`: A vector of parameters $[\nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$ without factor loadings.

- `GLweight`: A matrix of nodes and weights for Gauss-Legendre quadrature.
- `x_grid`: A vector of x values at which the CDF and PDF are evaluated.
- `lam_grid`: A vector of factor loading values at which the CDF and PDF are evaluated.

Output:

- `Gcdf`: A matrix representing the marginal CDFs of the skew t-t factor model at x and factor loading (lam).
- `Gpdf`: A matrix representing the marginal PDFs of the skew t-t factor model at x and factor loading (lam).

5.16 Function: X_pdf

Description: This function is used to calculate the marginal probability density function (PDF) of X_i associated with a factor model. It uses the `GLquad` function to numerically integrate the PDF of the innovations.

Usage:

```
out1 = X_pdf(x, theta, GLweight)
```

Input:

- `x`: A scalar, the value of x at which the PDF is evaluated.
- `theta`: A vector of parameters $[lam, \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$ representing the factor model.
- `GLweight`: A matrix of nodes and weights for Gauss-Legendre quadrature.

Output:

- `out1`: A scalar, the value of the marginal PDF at x.

5.17 Function: X_pdf_eval

Description: This function is used to evaluate the (approximate) marginal probability density functions (PDFs) at a vector of values q associated with a factor model.

It uses a spline interpolation on the pre-computed marginal PDFs (Gpdf) evaluated at `x_grid` and selects the appropriate PDF value based on the given `q`. If the spline interpolation fails, it falls back to numerical integration using `GLquad` and `X_pdf_helper`.

Usage:

```
out = X_pdf_eval(q, theta, Gpdf, x_grid, lam_grid)
```

Input:

- q: A vector of values at which the marginal PDFs are (approximately) evaluated.
- theta: A vector of parameters $[factor\ loading, \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$ representing the factor model.
- Gpdf: A matrix of the marginal PDFs of the skew t-t factor model at x and factor loading (lam).
- x_grid: A vector of x values at which Gpdf is evaluated.
- lam_grid: A vector of factor loading values at which Gpdf is evaluated.

Output:

- out: A vector, the (approximate) marginal PDFs evaluated at q.

5.18 Function: X_pdf_helper

Description: This helper function is used to calculate the argument of the integral for the calculation of the marginal probability density function (PDF) of X_i associated with a factor model.

Usage:

```
out1 = X_pdf_helper(u, x, theta)
```

Input:

- u: A $K \times 1$ vector of values of u used by the numerical integration function.
- x: A scalar, the value of x at which the argument of the integral is evaluated.
- theta: A vector of parameters $[lam, \nu_z^{-1}, \nu_\epsilon^{-1}, \psi_z]$ representing the factor model.

Output:

- out1: A $K \times 1$ vector, the value of the argument of the integral at each value of u.

5.19 Function: fminsearchbnd3

Description: This function is an extension of the fminsearch function in MATLAB, but with the addition of bound constraints by transformation. It performs unconstrained optimization of a given objective function, subject to lower and upper bounds on the variables. The optimization is performed using the Nelder-Mead simplex method.

Usage:

```
[x, fval, exitflag, output] = fminsearchbnd3(fun, x0, LB, UB
    , options, varargin)
```

Input:

- `fun`: The objective function to be minimized.
- `x0`: The initial guess for the optimal solution.
- `LB`: The lower bound vector or array. If no lower bounds exist for a variable, use `-inf`. If no lower bounds at all, `LB` may be left empty.
- `UB`: The upper bound vector or array. If no upper bounds exist for a variable, use `+inf`. If no upper bounds at all, `UB` may be left empty.
- `options`: Optional optimization parameters. If not provided, default options are used.
- `varargin`: Additional arguments to be passed to the objective function `fun`.

Output:

- `x`: The optimal solution.
- `fval`: The value of the objective function at the optimal solution.
- `exitflag`: An integer indicating the exit condition:
  - `1`: Function converged to a solution.
  - `0`: Maximum number of function evaluations reached.
- `output`: A structure containing information about the optimization process:
  - `iterations`: The number of iterations performed.
  - `funcount`: The number of function evaluations.
  - `algorithm`: The name of the optimization algorithm used (in this case, `fminsearch`).
  - `message`: A message indicating the termination reason.

5.20 Function: skewtdis_cdf

Description: This function calculates the cumulative distribution function (CDF) of the skewed t distribution proposed by Hansen (1994). The skewed t distribution is a generalization of the Student's t distribution with an additional skewness parameter.

Usage:

```
cdf = skewtdis_cdf(x, nu, lambda)
```

Input:

- `x`: A matrix, vector, or scalar of values at which to evaluate the CDF.
- `nu`: A matrix or scalar representing the degrees of freedom parameter of the skewed t distribution.
- `lambda`: A matrix or scalar representing the skewness parameter of the skewed t distribution.

Output:

- `cdf`: A matrix of the same size as `x`, containing the CDF values at each element of `x`.

Note: This function relies on the `tcdf` function, which calculates the CDF of the Student's t distribution.

References: This function was originally part of the "Spatial Econometrics" toolbox by James P. LeSage.

5.21 Function: skewtdis_inv

Description: This function calculates the inverse cumulative distribution function (CDF) of the skewed t distribution proposed by Hansen (1994). The skewed t distribution is a generalization of the Student's t distribution with an additional skewness parameter.

Usage:

```
inv = skewtdis_inv(u, nu, lambda)
```

Input:

- `u`: A matrix, vector, or scalar of values in the unit interval at which to evaluate the inverse CDF.
- `nu`: A matrix or scalar representing the degrees of freedom parameter of the skewed t distribution.
- `lambda`: A matrix or scalar representing the skewness parameter of the skewed t distribution.

Output:

- `inv`: A matrix of the same size as `u`, containing the inverse CDF values at each element of `u`.

Note: This function relies on the `tinv` function, which calculates the inverse CDF of the Student's t distribution.

References: This function was originally part of the "Spatial Econometrics" toolbox by James P. LeSage.

5.22 Function: skewtdis_pdf

Description: This function calculates the probability density function (PDF) of the skewed t distribution proposed by Hansen (1994). The skewed t distribution is a generalization of the Student's t distribution with an additional skewness parameter.

Usage:

```
pdf = skewtdis_pdf(x, nu, lambda)
```

Input:

- `x`: A matrix, vector, or scalar at which to evaluate the PDF.
- `nu`: A matrix or scalar representing the degrees of freedom parameter of the skewed t distribution.
- `lambda`: A matrix or scalar representing the skewness parameter of the skewed t distribution.

Output:

- `pdf`: A matrix of the same size as `x`, containing the PDF values at each element of `x`.

References: This function was originally part of the "Spatial Econometrics" toolbox by James P. LeSage.

5.23 Function: skewtdis_rnd

Description: This function generates random draws from the skewed t distribution proposed by Hansen (1994).

Usage:

```
out1 = skewtdis_rnd(nu, lambda, T, state)
```

Input:

- `nu`: A matrix or scalar representing the degrees of freedom parameter of the skewed t distribution.
- `lambda`: A matrix or scalar representing the skewness parameter of the skewed t distribution.
- `T`: The number of random draws to generate.
- `state`: (Optional) An integer used to seed the random number generator. If not provided, the generator will be seeded based on the current time.

Output:

- `out1`: A Tx1 vector containing the random draws from the skewed t distribution.

References: This function was originally part of the "Spatial Econometrics" toolbox by James P. LeSage.